**First Value Moment (FVM) - Cursor Implementation Prompts**

**Overview**

These prompts will create a comprehensive "First Value Moment" feature that provides immediate insights to new users, demonstrating Mingus's unique value proposition by combining financial, health, and important dates data.

**PROMPT 1: Create First Value Moment Service (Backend Core)**

Create a new file backend/services/first\_value\_moment\_service.py that integrates with the existing Mingus architecture.

Build a FirstValueMomentService class with these methods:

1. generate\_fvm\_insights(user\_id) -> dict with all three core insights

2. \_aggregate\_user\_data(user\_id) -> consolidated user data from all sources

3. \_validate\_data\_completeness(user\_data) -> ensure minimum data for insights

Data sources to integrate:

- UserProfile (from backend.models.user)

- EncryptedFinancialProfile (from backend.models.encrypted\_models)

- UserHealthCheckin (from backend.models.health)

- Important dates (from user preferences or financial goals)

- Existing UnifiedInsightsService

- Existing HealthCorrelationService

- Existing CashFlowAnalysisService

Requirements:

- Follow existing Mingus service patterns

- Include proper error handling for missing data

- Add logging for debugging

- Return structured JSON that frontend can consume

- Handle encrypted financial data properly

Make it compatible with the existing Flask app factory and service initialization patterns.

**PROMPT 2: Build Cash Flow Stress Analysis Method**

Add a method to FirstValueMomentService called analyze\_cash\_flow\_stress() that creates the first core insight.

Method signature:

analyze\_cash\_flow\_stress(user\_data) -> dict

This method should:

1. Generate 90-day cash flow projection using existing CashFlowAnalysisService

2. Identify the date with lowest projected balance

3. Correlate low balance dates with user's reported stress levels from health check-ins

4. Generate specific remedies based on user's current wellness habits

Output format:

{

"lowest\_balance\_date": "2025-08-15",

"lowest\_amount": 247,

"stress\_level": "high|medium|low",

"stress\_correlation": "Your stress peaks align with cash shortfalls",

"remedy": {

"current\_habit": "meditation: 5 minutes daily",

"suggestion": "increase to 15 minutes daily",

"potential\_impact": "save $85/month based on similar users"

},

"chart\_data": [array of daily balances for visualization]

}

Integration requirements:

- Use existing cash flow calculation logic

- Apply health correlation data from HealthCorrelationService

- Include demographic assumptions for target market (African Americans 25-35, $40k-$100k income)

- Handle missing health data gracefully

**PROMPT 3: Build Important Dates Reality Check Method**

Add a method to FirstValueMomentService called calculate\_dates\_affordability() that creates the second core insight.

Method signature:

calculate\_dates\_affordability(user\_data, cash\_flow\_projection) -> dict

This method should:

1. Extract important dates from user's financial goals, preferences, or manually entered dates

2. For each date, check if projected cash flow can cover the estimated cost

3. Calculate what percentage of dates are affordable

4. Determine weekly savings needed to afford all dates

5. Prioritize dates by importance/cost ratio

Output format:

{

"total\_dates": 5,

"affordable\_dates": 3,

"coverage\_percentage": 60,

"dates\_breakdown": [

{

"name": "Birthday dinner",

"date": "2025-09-15",

"estimated\_cost": 150,

"projected\_balance": 890,

"affordable": true,

"confidence": "high"

},

{

"name": "Anniversary trip",

"date": "2025-10-22",

"estimated\_cost": 800,

"projected\_balance": 320,

"affordable": false,

"shortfall": 480

}

],

"solution": {

"weekly\_savings\_needed": 47,

"timeline": "8 weeks to full coverage",

"priority\_adjustment": "Consider reducing anniversary trip cost to $600"

}

}

Requirements:

- Integrate with existing financial goals system

- Use realistic cost estimates based on user location (target market cities)

- Handle cases where user hasn't entered specific dates

- Provide actionable solutions, not just problems

**PROMPT 4: Build Wellness-Spending Prediction Method**

Add a method to FirstValueMomentService called predict\_wellness\_spending\_impact() that creates the third core insight.

Method signature:

predict\_wellness\_spending\_impact(user\_data, health\_correlations) -> dict

This method should:

1. Analyze user's current wellness habits from health check-in data

2. Use existing health-spending correlations to predict current overspending

3. Generate specific improvement scenarios with projected savings

4. Base predictions on target demographic data (African Americans, age 25-35, target cities)

Output format:

{

"current\_assessment": {

"stress\_level": "high",

"exercise\_frequency": "low",

"meditation\_minutes": 5,

"relationship\_satisfaction": "medium",

"predicted\_stress\_spending": 127

},

"improvement\_scenarios": [

{

"type": "meditation",

"current": "5 minutes daily",

"suggested": "15 minutes daily",

"potential\_monthly\_savings": 85,

"confidence": "high",

"rationale": "Users with similar profiles who meditate 15+ min save avg $85/month"

},

{

"type": "exercise",

"current": "1 workout/week",

"suggested": "3 workouts/week",

"potential\_monthly\_savings": 62,

"confidence": "medium",

"rationale": "Regular exercise reduces impulse spending by avg $62/month"

}

],

"total\_potential\_savings": 147,

"recommendation": "Start with meditation - easier habit with quick financial impact"

}

Requirements:

- Use existing HealthCorrelationService data

- Apply culturally relevant assumptions for target market

- Provide realistic, achievable suggestions

- Include confidence levels based on data quality

**PROMPT 5: Create FVM API Endpoint**

Create a new API endpoint in the existing Flask route structure for the First Value Moment feature.

Create or modify: backend/routes/insights\_routes.py

Add endpoint:

@blueprint.route('/api/insights/first-value-moment', methods=['GET'])

The endpoint should:

1. Get current authenticated user ID

2. Call FirstValueMomentService.generate\_fvm\_insights(user\_id)

3. Handle cases where user doesn't have enough data yet

4. Return appropriate HTTP status codes and error messages

5. Follow existing Mingus API patterns for response formatting

Response format for success:

{

"success": true,

"data": {

"cash\_flow\_analysis": {...},

"dates\_reality\_check": {...},

"wellness\_predictions": {...},

"generated\_at": "2025-07-16T10:30:00Z",

"data\_completeness": 85

}

}

Response format for insufficient data:

{

"success": false,

"error": "insufficient\_data",

"message": "Please complete your financial profile and first health check-in",

"missing\_requirements": ["income\_data", "health\_checkin"],

"next\_steps": "/api/onboarding/next-step"

}

Requirements:

- Integrate with existing authentication middleware

- Add proper audit logging using existing AuditService

- Include rate limiting for the endpoint

- Follow existing error handling patterns

- Make it compatible with existing route registration in app factory

**PROMPT 6: Build React FVM Dashboard Component**

Create a new React component: src/components/FirstValueMoment.jsx

This component should display the three core insights in an engaging, visual way:

1. Cash Flow Stress Analysis:

- Line chart showing 90-day cash flow projection

- Color-coded stress zones (red for danger periods)

- Highlighted lowest balance date

- Remedy suggestion card with specific action

2. Important Dates Reality Check:

- Progress circle showing percentage of affordable dates

- List of specific dates with checkmarks/X marks

- Weekly savings goal prominently displayed

- Timeline showing when full coverage is achievable

3. Wellness-Spending Predictions:

- Current habits summary cards

- Before/after scenarios with dollar amounts

- Interactive "What if" sliders for habit improvements

- Clear next action recommendations

Technical requirements:

- Use existing Mingus design system and styling

- Make fully responsive for mobile users

- Include loading states while data fetches

- Handle error states gracefully

- Add smooth animations for engagement

- Include clear CTAs for next steps

- Follow existing React patterns used in Mingus

Data fetching:

- Use existing API service patterns

- Implement proper error handling

- Add retry logic for failed requests

- Cache insights data for performance

The component should feel like a personalized financial health report that gets users excited about the app's capabilities.

**PROMPT 7: Create FVM Integration and Navigation Flow**

Modify the existing onboarding flow to include the First Value Moment at the right point.

Files to modify:

- src/components/onboarding/ (existing onboarding components)

- backend/services/onboarding\_service.py

- backend/routes/onboarding\_routes.py

Integration requirements:

1. Trigger FVM generation after user completes:

- Basic financial profile (income, expenses)

- First health check-in (stress, exercise, meditation)

- At least 2 important dates entered

2. Add FVM step to onboarding flow:

- New step: "See Your Financial Health Snapshot"

- Position it as the "reveal" moment after data collection

- Include option to skip for users who want to explore first

3. Create navigation helpers:

- Add "Show My Insights" button throughout the app

- Include FVM summary card on main dashboard

- Add sharing functionality for insights

4. Update onboarding progress tracking:

- Mark FVM completion as onboarding milestone

- Track user engagement with each insight type

- Record which insights lead to specific actions

5. Add contextual help:

- Tooltips explaining each insight

- "Learn more" links to relevant app features

- Guided tour highlighting how to improve each metric

Requirements:

- Maintain existing onboarding progress tracking

- Follow existing React Router patterns

- Add proper analytics tracking for insight engagement

- Include A/B testing hooks for future optimization

- Make FVM accessible from multiple entry points in the app

**PROMPT 8: Add FVM Database Models and Migrations**

Create database models to track First Value Moment generation and user engagement.

Create new file: backend/models/fvm\_models.py

Models needed:

1. FVMGeneration:

- user\_id (foreign key)

- generated\_at (timestamp)

- data\_completeness\_score (0-100)

- insights\_generated (JSON blob)

- user\_satisfaction\_rating (1-5, nullable)

2. FVMEngagement:

- fvm\_generation\_id (foreign key)

- insight\_type (cash\_flow, dates, wellness)

- action\_taken (viewed, shared, acted\_on)

- time\_spent\_seconds

- engagement\_at (timestamp)

3. FVMFeedback:

- fvm\_generation\_id (foreign key)

- feedback\_type (helpful, confusing, inaccurate)

- specific\_insight (which insight the feedback relates to)

- comment (text, nullable)

- submitted\_at (timestamp)

Create migration script: backend/migrations/add\_fvm\_tables.py

Requirements:

- Follow existing Mingus database patterns

- Include proper indexes for performance

- Add foreign key constraints

- Include audit fields (created\_at, updated\_at)

- Make models compatible with existing SQLAlchemy setup

- Add proper model relationships

Also create service methods:

- log\_fvm\_generation()

- track\_insight\_engagement()

- collect\_user\_feedback()

- generate\_fvm\_analytics()

Integration with existing audit system for compliance tracking.

**PROMPT 9: Build FVM Analytics and Performance Monitoring**

Create analytics tracking for First Value Moment feature performance.

Create file: backend/analytics/fvm\_analytics.py

Build FVMAnalytics class with methods:

1. track\_generation\_success(user\_id, insights\_data, data\_completeness)

2. track\_user\_engagement(user\_id, insight\_type, action, time\_spent)

3. measure\_conversion\_impact(fvm\_users vs non\_fvm\_users)

4. generate\_insight\_performance\_report() -> which insights drive most engagement

5. identify\_improvement\_opportunities() -> where users get confused/disengage

Key metrics to track:

- FVM completion rate by user segment

- Time spent on each insight type

- User actions taken after viewing insights

- Correlation between FVM engagement and app retention

- Accuracy of predictions vs actual user behavior

- User satisfaction ratings by insight type

Create dashboard endpoint:

POST /api/analytics/fvm-performance

Return format:

{

"period": "last\_30\_days",

"total\_generations": 1247,

"completion\_rate": 84.2,

"engagement\_by\_insight": {

"cash\_flow": {"avg\_time": 45, "action\_rate": 72},

"dates": {"avg\_time": 32, "action\_rate": 89},

"wellness": {"avg\_time": 58, "action\_rate": 63}

},

"conversion\_impact": {

"trial\_to\_paid\_with\_fvm": 68.5,

"trial\_to\_paid\_without\_fvm": 42.1

},

"top\_improvement\_opportunities": [...]

}

Requirements:

- Integrate with existing analytics infrastructure

- Follow privacy compliance requirements

- Include A/B testing support for future optimization

- Create alerts for concerning metric drops

- Export data for business intelligence tools

**PROMPT 10: Create FVM Testing Suite**

Create comprehensive testing for the First Value Moment feature.

Create test files:

- tests/services/test\_first\_value\_moment\_service.py

- tests/routes/test\_fvm\_api.py

- tests/frontend/test\_fvm\_component.py

Test scenarios for FirstValueMomentService:

1. Happy path tests:

- User with complete data gets all three insights

- Insights are mathematically accurate

- Remedies are appropriate for user demographics

2. Edge case tests:

- User with minimal financial data

- User with no health check-ins yet

- User with no important dates entered

- User with extremely high/low income

3. Integration tests:

- Verify compatibility with existing services

- Test encrypted financial data handling

- Confirm audit logging works correctly

4. Performance tests:

- Insight generation completes under 3 seconds

- Handles multiple concurrent users

- Database queries are optimized

Frontend component tests:

- Component renders with sample data

- Loading states display correctly

- Error handling works properly

- Mobile responsiveness verified

- Accessibility compliance checked

API endpoint tests:

- Authentication required

- Proper HTTP status codes

- Rate limiting functions

- Data validation works

- Error responses are helpful

Create mock data generators:

- create\_test\_user\_with\_complete\_data()

- create\_test\_user\_with\_minimal\_data()

- generate\_sample\_fvm\_insights()

Requirements:

- Follow existing Mingus testing patterns

- Include both unit and integration tests

- Add performance benchmarks

- Create fixtures for consistent test data

- Include accessibility testing

- Add tests for all error conditions

**Implementation Timeline**

**Week 1: Backend Foundation (Days 1-5)**

* **Day 1**: Prompts 1-2 (Core service and cash flow analysis)
* **Day 2**: Prompt 3 (Important dates calculation)
* **Day 3**: Prompt 4 (Wellness predictions)
* **Day 4**: Prompt 5 (API endpoint)
* **Day 5**: Prompt 8 (Database models)

**Week 2: Frontend and Integration (Days 6-10)**

* **Day 6**: Prompt 6 (React component)
* **Day 7**: Prompt 7 (Navigation integration)
* **Day 8**: Prompt 9 (Analytics)
* **Day 9**: Prompt 10 (Testing)
* **Day 10**: Integration testing and bug fixes

**Success Criteria**

**Technical**

* All tests pass with >95% code coverage
* FVM generation completes in <3 seconds
* Frontend component renders properly on all device sizes
* API endpoint handles 100+ concurrent requests

**User Experience**

* Users spend 3+ minutes reviewing insights (target: 80%)
* Users click through to main app features (target: 75%)
* Users rate insights as helpful (target: 85%)

**Business Impact**

* Increase trial-to-paid conversion by 40-50%
* Improve 30-day retention by 35%+
* Generate referrals from impressed users

This comprehensive implementation plan creates a powerful First Value Moment that demonstrates Mingus's unique value proposition while building on your existing technical infrastructure.